On the Generalization Capabilities of Sharp Minima in Case-Based Reasoning

Thomas Gabel and Eicke Godehardt

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
{tgabel|godehardt}@fb2.fra-uas.de

Abstract. In machine learning and numerical optimization, there has been an ongoing debate about properties of local optima and the impact of these properties on generalization. In this paper, we make a first attempt to address this question for case-based reasoning systems, more specifically for instance-based learning as it takes place in the retain phase. In so doing, we cast case learning as an optimization problem, develop a notion of local optima, propose a measure for the flatness or sharpness of these optima and empirically evaluate the relation between sharp minima and the generalization performance of the corresponding learned case base.

1 Introduction

Powered by a number of recent empirical successes, the field of deep learning has become one of the most noticed sub-fields of artificial intelligence during the past few years. Training deep neural networks means solving a complex, non-convex optimization problem. Interestingly, gradient-based optimization of such networks takes place in an over-parameterized setting, where the target function has, in general, a vast number of local and multiple global optima. All of them minimize the train error, but typically many of them generalize poorly. Consequently, just minimizing the train error is merely adequate, since a poorly chosen minimum may bring about bad performance on independent test data. It has been generally accepted that the generalization capabilities do implicitly depend on the algorithm used for minimizing the train error – since that algorithm determines which minimum it gets attracted by – and it is an ongoing debate to which extent properties of the attained minimum can be indicative for generalization.

In this paper, we transfer these thoughts to case-based reasoning. After providing some background and pointers to related work in Section 2, we start by searching for a related (optimization) problem in CBR and find one in the retain phase where case base editing and maintenance can naturally be cast as a discrete NP-hard optimization problem. Put simply, the question addressed here is which cases from a given set of train cases shall be retained in the case base and which not, while optimizing some objective function. In Section 3, we

model this problem formally and, for measuring an edited case base's competence, we derive an error function around which we center our further analyses made in this paper. As a first contribution, we then develop four variants of two hill-climbing case base editing algorithms (FHC and MHC) which, by design, are attracted by a local optimum of the hitherto derived objective function. The second main contribution of this paper follows in Section 4. There, we aim at characterizing an edited case base configuration (and, hence, a possibly attained minimum in the error landscape) as being "flat" or "sharp". In so doing, we derive an appropriate measure of sharpness which, informally speaking, approximates gradient information in the near neighborhood of the edited case base to the extent that this is feasible in the given non-continuous optimization task. The last part of the paper (Section 5) is devoted to an empirical evaluation of our approach using a large set of established classification problems. To this end, our hypothesis is that there is some correlation between the sharpness of a case base configuration as defined before and the generalization capabilities of the resulting case-based classifier. A secondary objective also covered in our evaluation concerns the actual empirical power of the hill-climbing case base editing schemes that we proposed in Section 3.

2 Background and Related Work

We start by providing some general basics as well as a nearly chronological, brief survey on related work on issues that relate to maintaining case bases. We then adopt an optimization point of view and briefly introduce some foundations on function minimization including the notion of sharp and flat minima.

2.1 Case Base Maintenance and Instance-Based Learning

Case base maintenance (CBM [10]) is known for addressing two goals in case-based reasoning: (1) controlling the number of cases in the case base and, hence, avoiding a performance degradation due to outrageous growth and (2) assuring a high competence of the case base. As a matter of fact, CBM is probably that component of the CBR realm that bears the strongest[1] relation to learning in the classical machine learning sense. Issues of case base maintenance arise early in almost any CBR system which is why this sub-field of CBR has attracted so much attention during the past decades. Starting with the initial proposal of the nearest neighbor rule [3] several authors subsequently aimed at reducing the size of the set of stored instances [6, 5, 18]. Later, Aha and varying co-authors proposed the family of instance-based learning algorithms (IBL [1]) where new instances are stored subject to different criteria, as well as subtractive counterparts (e.g. [9]). Other approaches more intensively focused on the location of retained instances, such as preferably keeping those near the center of clusters

[1] Though learning can take place also in one of the other knowledge containers of a CBR system, e.g. when learning similarity measures or adaption knowledge.

rather than near to decision borders [20]. Another piece of work which – at least from the algorithm's name – suggests to be related to the algorithms we introduce in Section 3.4 seems to be random mutation hill-climbing [15]. However, as it turns out our algorithmic approach differs in various ways, most importantly as we do not fix the case base size beforehand, such that the remaining commonality is the hill-climbing nature of the procedure. The family of decremental reduction optimization procedures (DROP [19]) takes the opposite approach and iteratively decides which cases to delete from a case base without deteriorating competence (cf. Section 3.4.3 for more details on this). In another line of research, Smyth and varying co-authors approached the problem of case base size limitation using the notion of coverage and reachability of cases [16], concepts that were later exploited by the COV-FP [17] and ICF [2] algorithms. Finally, the idea that case base maintenance represents an optimization problem with multiple goals to be pursued simultaneously appears most pronounced in [14].

2.2 Sharp and Flat Minima of an Error Function

Given some real-valued function f over some domain X, f is said to have a minimum at $m \in X$, if there exists some $\varepsilon > 0$ such that $f(m) \leq f(x) \forall x \in X$ with $d_X(x, m) < \varepsilon$ where d_X measures the distance of x and m. The idea of characterizing minima as sharp or flat has received much attention in the literature especially in the context of the recent rise of deep learning systems [8, 4, 12], but dating back at least to the seminal work of Schmidhuber [7]. A flat minimum m_f is said to be a point where the function f varies slowly in a relatively large neighborhood of m_f. By contrast, a sharp minimum m_s is a point where the function f increases rather strongly in the near vicinity of m_s. While exact numerical values for measuring the sharpness of a minimum would strongly depend on the scaling of the function f and its inputs as well as on the definition of what a large neighborhood or near vicinity means, it is intuitive that the large sensitivity of a sharp minimum (little changes to m_s yield strong changes to f) may negatively impact the generalization capabilities of the system. Even if an exact numerical score of the sharpness of a minimum may not be as informative in itself, it still may be important when selecting between different minima, e.g. two minima m_1 and m_2 with $f(m_1) = f(m_2)$, but with strongly different levels of sharpness. Figure 1 aims at visualizing this issue conceptually in a one-dimensional space [8]: The function f to be minimized (as it may stem from a set of given training samples) is shown with a solid line, whereas the ground truth g, i.e. the true relation to be learned as it may be represented by a (large) independent set of test data, is depicted with a dashed line. While both, the flat and the sharp minimum have the same training performance, the testing performance, i.e. the generalization capability, of m_s is much worse.

3 Case Base Maintenance as Optimization Problem

In what follows, we aim at an in-depth analysis of what it means for a CBR system to "learn cases" (or to delete some), bridging the gap to issues such as

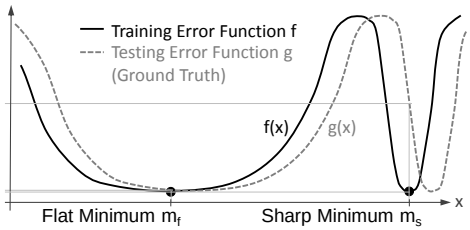

Fig. 1. Visualization of Sharp vs. Flat Minima: The abscissa shows the domain of the search space, the ordinate shows the value of the error function.

optimization, minima found during optimization, generalization, and analyzing the implications for (more or less) established CBM strategies.

3.1 Case Base Editing Problem

Let us denote the set of all cases as $\mathcal{M}$ and, as usual, each case $c = (p, s) \in \mathcal{M}$ be composed of a problem part $p \in P$ and a solution part $s \in S$, i.e. $\mathcal{M} = P \times S$. Now assume, we are given a set $\mathcal{T} \subset \mathcal{M}$ of training cases. Then, the case base editing problem we are focusing on means finding a case base $\mathcal{S}$ as a subset of $\mathcal{T}$ that features as much of the following properties as possible:

- $\mathcal{S} \subset \mathcal{T}$, i.e. $\mathcal{S}$ should be a (desirably small) subset of $\mathcal{T}$ for reasons of retrieval efficiency
- $\mathcal{S}$ should be as competent as possible where competence is typically measured as its problem-solving capability on a disjoint set $\mathcal{U} \subset \mathcal{M}$ of test cases (i.e. $\mathcal{T} \cap \mathcal{U} = \emptyset$)

From a global point of view, this setting gives rise to $2^{|\mathcal{T}|}$ possible configurations for the resulting case base $\mathcal{S}$ since each $c \in \mathcal{T}$ can be either contained or not contained in $\mathcal{S}$, yielding the power set $\mathfrak{P}(\mathcal{T})$ of $\mathcal{T}$ as the search space. Needless to say, that *any* CBM strategy described in the literature performs some kind of search through that space $\mathfrak{P}(\mathcal{T})$, being guided either heuristically or by certain performance criteria.

Definition 1 (Case Base Configuration). *Given training cases $\mathcal{T}$ any non-empty subset $\mathcal{S} \subset \mathcal{T}$ represents a valid* case base configuration *(CBC) in the context of case base editing, i.e. $\mathcal{S} \in \mathfrak{P}(\mathcal{T})$.*

Accordingly, searching the whole space of all case base configurations is intractable in general, except for toy problems. However, this setting allows us to develop an intuition for local versus non-local changes to a case base.

Definition 2 (Case Editing Operator). *Given some case base configuration $\mathcal{S}$ and a case $c \in \mathcal{T}$, the* case editing operator $\mathcal{E} : \mathfrak{P}(\mathcal{T}) \times \mathcal{T} \to \mathfrak{P}(\mathcal{T})$ *returns a new case base configuration $\mathcal{S}'$ such that*

$$\mathcal{S}' = \begin{cases} \mathcal{S} \setminus \{c\} & \text{if } c \in \mathcal{S} \\ \mathcal{S} \cup \{c\} & \text{else} \end{cases}$$

Clearly, the change that $\mathcal{E}$ introduces to $\mathcal{S}$ is the smallest one possible – we could also say, it is a local change to $\mathcal{S}$ –, since only the membership of a single case c is swapped. By contrast, any case base $\mathcal{S}_1$ can be changed to any $\mathcal{S}_2$ by applying a sequence of such atomic operations where the length of that sequence is, informally, given by the Hamming distance of $\mathcal{S}_1$ and $\mathcal{S}_2$. Note that Definition 2 is not tailored to data sets where a case is contained multiple times in $\mathcal{S}$.

3.2 Introspective Problem-Solving Quality

Different authors have employed different measures for the problem-solving quality of an edited case base. For example, Lupiani et al. [14] define a multi-objective error function combining error, noise, and redundancy which is to be minimized by an evolutionary algorithm, while Smyth and McKenna [16] center competence around the notion of coverage and reachability. A widespread measure for estimating the problem-solving capability of a case base $\mathcal{S}$ is the leave-one-out error (or accuracy) where each case c is used as query once using $\mathcal{S} \setminus \{c\}$ as leave-one-out case base (LOO [19, 11]). Throughout the rest of this paper, we stick to a slight modification of this established measure for case base competence due to its simplicity and due to the fact that no further knowledge is needed (e.g. for generating sample solutions) which eases the empirical evaluation.

Definition 3 (Leave-One-Out Train Error of a Case Base Configuration). *For a training set of cases $\mathcal{T} = \{c_1, \ldots, c_{|\mathcal{T}|}\}$ with each case $c_i = (p_i, s_i)$ consisting of a problem and solution part and for a given case base configuration $\mathcal{S}$, the* leave-one-out train error *is defined as*

$$\mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S}) = \frac{1}{|\mathcal{T}|} \cdot \sum_{i=1}^{|\mathcal{T}|} 1 - Correct(Adapt(Retrieve(\mathcal{S} \setminus \{c_i\}, p_i), p_i), s_i) \qquad (1)$$

In that definition, the *Retrieve* function performs case-based retrieval over the leave-one-out case base $\mathcal{S} \setminus \{c_i\}$ using p_i as query. To this end, no restrictions on the retrieval algorithm or the used similarity measures or the value of k in case of a k-nearest neighbor retrieval are made. The *Adapt* function takes the set of nearest neighbors returned by the retrieval and performs adaptation to form a single unique suggested solution or does nothing, if no adaption knowledge is available or necessary. Finally, that returned solution is checked against the solution s_i of case c_i whose problem part p_i was used as query in the first place. If function *Correct* finds that both solutions are identical (or sufficiently identical), it returns 1, otherwise 0. In fact, we will focus on classical classification domains in the remainder of this paper such that indeed no adaptation will be performed and the correctness check is simplified to the matching of class labels. Also note, the small difference to the standard LOO definition for case base competence is that our measure iterates not just over the case base itself, but over all cases in

the training set expecting to have a larger sample of the entire problem space $\mathcal{M}$. This is in compliance with the representative assumption for the competence of case bases first proposed in [16].

3.3 Local Optima in Case Base Editing

The search space $\mathfrak{P}(\mathcal{T})$ of case base configurations comprises $|\mathcal{T}|$ dimensions, along each of which only two values are possible (case is "in" or "out"). As a consequence, a case base configuration $\mathcal{S}$ is a *local optimum* (minimum) in the error landscape of $\mathfrak{P}(\mathcal{T})$, if

$$\forall c \in \mathcal{T} : \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S}) \leq \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{E}(\mathcal{S}, c)) \tag{2}$$

since $\mathcal{E}(\mathcal{S}, c)$ on the right-hand side refers to a case base that represents a minimal adaptation to $\mathcal{S}$. Accordingly, a strict minimum is attained, if we replace the less-equal sign by a strict less.

Certainly, $\mathfrak{P}(\mathcal{T})$ is full of configurations $\mathcal{S}$ where switching on/off a single case no further reduces the value of the error; in that case $\mathcal{S}$ is a local minimum. Switching on/off a number of cases simultaneously might, however, still bring about improvements. This is exactly what Lupiani et al. [14] are exploiting using evolutionary algorithms where a "more global" search through $\mathfrak{P}(\mathcal{T})$ can be done using crossover. By contrast, the hunt for a global optimum (or a nearly optimal local one) is not our primary concern in this paper. Instead, we are more interested in characterizing the properties of different local minima. Therefore, in the next section, we suggest a number of case base editing schemes, that are based on $\mathbb{E}^{loo}_{\mathcal{T}}$ and that will, by definition, find various local minima easily.

3.4 Hill-Climbing Case Base Editors

For analyzing the properties of local minima in case base editing more thoroughly, it is comfortable to have access to a way for generating such optima easily. Thus, we suggest a set of greedy algorithms for case base editing, which are – because they are hill-climbers – designed to converge to local optima in the error landscape quickly. Besides, we also review a set of well-established case base editing methods from the literature and discuss whether they yield local optima of $\mathbb{E}^{loo}_{\mathcal{T}}$ as well. We developed these algorithms mainly for reasons of analyzing sharp/flat minima of the error function, being aware that they are unlikely to yield a global optimum in the search landscape (unlike e.g. [14]). However, their empirical performance matches up to the performance of a number of established CBM methods that we implemented for the purpose of further analysis.

3.4.1 First Improvement Hill-Climber (FHC)

This algorithm is called $FHC_<$ and, similarly as IB2 or CNN (cf. Section 3.4.3), starts out with an empty case base configuration $\mathcal{S} = \emptyset$. It then iterates over all cases c in $\mathcal{T}$ and checks for the first case to fulfill the following condition:

$$\mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{E}(\mathcal{S}, c)) < \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S}), \tag{3}$$

i.e. the first case whose addition to or removal from the current case base configuration $\mathcal{S}$ reduces the leave-one-out train error over $\mathcal{T}$. If such a case c is found, the case editing operator $\mathcal{E}$ either adds or removes c to/from $\mathcal{S}$ (depending on whether it was already contained or not). This procedure is repeated until there is no more case in $\mathcal{T}$ for which Equation 3 becomes true. Algorithm 1 shows pseudo-code for an implementation of $FHC_<$.

Input: training set $\mathcal{T} \subset \mathcal{M}$, **Output:** case base configuration $\mathcal{S}$ ($\mathcal{S} \subseteq \mathcal{T}$)

Strict variant $FHC_<$

```
1:  S ← ∅, stop ← false
2:  while stop = false do
3:      stop ← true
4:      for c ∈ T do
5:          if E^loo_T(E(S,c)) < E^loo_T(S)
6:          then
7:              S ← E(S,c)
8:              stop ← false
9:              quit for loop
10: return S
```

Modification for non-strict variant $FHC_\leq$

```
4: ...
5: if E^loo_T(E(S,c)) < E^loo_T(S) or
       (E^loo_T(E(S,c)) = E^loo_T(S) and c ∉ S)
6: then ...
```

Algorithm 1: First Improvement Hill-Climber (variants $FHC_<$ and $FHC_\leq$)

A variation of $FHC_<$ is attained, if we replace the strict inequality in Equation 3 by a less-equal comparison; this variant is named $FHC_\leq$ accordingly. Clearly, $FHC_\leq$ will in general add more cases to the case base than $FHC_<$. It is also obvious that both variants will, due to their hill-climbing nature, be attracted by a local optimum in the error landscape according to Equation 3. From an algorithmic point of view, one should take care that $FHC_\leq$ does not get trapped in an endless loop due to the addition/removal of some "irrelevant" case whose presence/absence does not alter $\mathbb{E}^{loo}_{\mathcal{T}}$. To this end, we decided to allow for a non-strict comparison for the addition of a case and retain a strict change of the error for removing a case (see right part of Algorithm 1).

3.4.2 Maximum Improvement Hill-Climber (MHC)

The main difference between $FHC_<$ and the maximum improvement hill-climber $MHC_<$ introduced next lies in the selection of the next case that is added to or removed from the current case base configuration $\mathcal{S}$. While $FHC_<$ picks the first case $c \in \mathcal{T}$ whose de/activation brings about an improvement of the train error $\mathbb{E}^{loo}_{\mathcal{T}}$, $MHC_<$ performs an entire sweep over $\mathcal{T}$ and selects that c^* that yields the largest improvement.

So, while the pseudo-code in Algorithm 2 seems quite compact, it hides part of its complexity in the arg min operator in line 4. Despite having a larger complexity in its outer *while* loop, $MHC_<$ will on average terminate faster than $FHC_<$ as it requires less iterations of that outer *while* loop because each one yields the maximal possible reduction of the train error. Consequently, it tends

to create smaller case bases and terminate faster. Again, a less strict variant of $MHC_<$ is realized, if the inequality is replaced by a less-equal in line 5 (called $MHC_\leq$). Finally, it is trivial to see that both MHC variants do always end up in a local minimum of the error landscape of $\mathbb{E}^{loo}_{\mathcal{T}}$. As a side note we remark that MHC is nearly[2] insensitive to the "presentation order" of cases which is a criticism to most of the established case base maintenance algorithms [14].

Input: training set $\mathcal{T} \subset \mathcal{M}$, **Output:** case base configuration $\mathcal{S}$ ($\mathcal{S} \subseteq \mathcal{T}$)

Strict variant $MHC_<$	*Modification for non-strict variant* $MHC_\leq$
1: $\mathcal{S} \leftarrow \emptyset$, $stop \leftarrow false$	
2: **while** $stop = false$ **do**	
3: $stop \leftarrow true$	
4: $c^* \leftarrow \arg\min_{c \in \mathcal{T}} \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{E}(\mathcal{S}, c))$	4: ...
5: **if** $\mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{E}(\mathcal{S}, c^*)) < \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S})$	5: **if** $\mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{E}(\mathcal{S}, c^*)) < \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S})$ **or**
6: **then**	$(\mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{E}(\mathcal{S}, c^*)) = \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S})$ **and** $c \notin \mathcal{S})$
7: $\mathcal{S} \leftarrow \mathcal{E}(\mathcal{S}, c^*)$	6: **then** ...
8: $stop \leftarrow false$	
9: **return** $\mathcal{S}$	

Algorithm 2: Maximum Improvement Hill-Climber ($MHC_<$ and $MHC_\leq$)

3.4.3 Related Case Base Editing Schemes

As mentioned, we incorporate a set of well-known case base editing methods into our further analyses. We highlight each of them with some remarks, emphasizing that this list is not complete and could easily be extended by many further algorithms from the realm of case base maintenance.

CNN (condensed nearest neighbor [6]) might be called one of the forefathers of CBM. It starts with an empty case base, makes multiple passes over $\mathcal{T}$ and copies a case c from $\mathcal{T}$ to $\mathcal{S}$, if it finds that c cannot be solved by $\mathcal{S}$. CNN has inspired various alternative and more sophisticated case base editing rules. In its original form it aims at finding a subset $\mathcal{S}$ of $\mathcal{T}$ that is as consistent as $\mathcal{T}$. Formally, CNN minimizes, starting with an empty case base configuration, an error function that is similar to $\mathbb{E}^{loo}_{\mathcal{T}}$, but not defined in a leave-one-out manner. As a consequence and due to the fact that CNN can just add cases to $\mathcal{S}$ and not remove them (like FHC or MHC), the output of CNN will in general not correspond to a local minimum of $\mathbb{E}^{loo}_{\mathcal{T}}$ as defined above.

RNN is an extension of the aforementioned CNN which adds a case removal phase during which cases are deleted from $\mathcal{S}$ whose removal does not impair the problem-solving competence of $\mathcal{S}$ on all cases from $\mathcal{T}$ [5]. The output of RNN might in general be expected to be closer to an optimum of $\mathbb{E}^{loo}_{\mathcal{T}}$ than

[2] There remains some sensitivity to the presentation order since in line 4 multiple cases c may reduce $\mathbb{E}^{loo}_{\mathcal{T}}$ equally in which case one of those cases must be selected, e.g. randomly or by some convention.

CNN's output. However, RNN is not an optimizer of $\mathbb{E}_{\mathcal{T}}^{loo}$ since case addition and removal are strictly split into two separate phases and because the error is not measured in a leave-one-out manner.

IBL Algorithms denote instance-based learning algorithms [1]. IB2, as one instance of this family of algorithms, iterates over $\mathcal{T}$ and adds a case c to $\mathcal{S}$, if c's problem part would not be solved correctly by the cases in $\mathcal{S}$. As a consequence, it is susceptible to noise, but it can also be termed a greedy algorithm in the sense that it tries to add as little cases as possible.

DROP Algorithms denote decremental reduction optimization procedures [19]. Different variants exist; DROP1 starts out with a case base $\mathcal{S}$ that is set to the full set of train cases, $\mathcal{S} = \mathcal{T}$. Then, it iteratively removes individual cases from $\mathcal{T}$ whose deletion does not worsen the leave-one-out performance over $\mathcal{S}$ (note, not over $\mathcal{T}$). DROP2 is an extension of DROP1 whose leave-one-out performance is measured over the whole set $\mathcal{T}$. Insofar, DROP2 comes close to our FHC and MHC algorithms, except for its inability to re-add cases after having deleted them. Moreover, DROP2 employs also a specialized preference heuristic regarding which cases to remove first, namely those which have the smallest similarity to their "nearest enemy" (which means a case in $\mathcal{T}$ whose solution does not match or cannot be adapted). As a consequence, DROP2 is more likely than all other algorithms listed here to yield a local optimum in the sense that we defined in Section 3.3.

4 Sharpness of a Case Base Configuration

In the preceding section, we have formalized case base editing as an optimization problem where we (a) defined an error measure $\mathbb{E}$ over the training set that relates to the leave-one-out competence of the system over a training set $\mathcal{T}$ and (b) proposed discrete editing operations (case editing operator $\mathcal{E}$) for searching for a minimum of $\mathbb{E}_{\mathcal{T}}^{loo}$. All these steps were necessary to path the way for a further analysis of local optima in the error landscape that we are intending to describe now. Additionally, for performing the actual search, we suggested two hill-climbing algorithms (FHC and MHC, more specifically four variants of them) which are, by definition, designed to find local optima for $\mathbb{E}$.

4.1 Characterizing Flat and Sharp Case Base Editing Optima

As described in Section 2.2, a sharp minimum m_s of some function f over domain X is characterized by the observation that f changes rapidly in the near vicinity o m_s. We have also highlighted that in related research fields sharp minima are known to correspond to models with poor generalization capabilities.

The case base editing problem, as defined in Section 3.1 is, however, of discrete nature. Instead of the mentioned numeric domain X, for a given set of training cases $\mathcal{T}$, the search space contains the $2^{|\mathcal{T}|}$ elements of $\mathcal{T}$'s power set $\mathfrak{P}(\mathcal{T})$. One might say, the space to be searched is $|\mathcal{T}|$-dimensional with two possible values in each dimension. Acknowledging this and aiming at establishing a notion of the vicinity around some minimum, we thus define:

Definition 4 (Vicinity of a Case Base Configuration). *Given a set of training cases $\mathcal{T}$ and a case base configuration $\mathcal{S}$ with $\mathcal{S} \subseteq \mathcal{T}$, the* vicinity of $\mathcal{S}$ *is defined as*

$$\mathcal{V}_{\mathcal{T}}(\mathcal{S}) = \{\mathcal{E}(\mathcal{S}, c) | c \in \mathcal{T}\}$$

Hence, the vicinity of a case base configuration contains all $|\mathcal{T}|$ case bases that are formed, if we either leave out exactly one $c \in \mathcal{S}$ from $\mathcal{S}$ or if we add exactly one case from $\mathcal{T} \setminus \mathcal{S}$ to $\mathcal{S}$. If $\mathcal{S}$ is known to be a local optimum (e.g. as the result of applying FHC or MHC, cf. Section 3.4), then by construction it holds that the leave-one-out train error for $\mathcal{S}$ is smaller than or equal to the error for any case base configuration within the vicinity set $\mathcal{V}(\mathcal{S})$.

The vicinity definition allows us to derive a numeric estimation of the sharpness of some case base configuration – a notion that of course covers local optima of case base editing as well.

Definition 5 (Sharpness of a Case Base Configuration). *Given a set of training cases $\mathcal{T}$ and a case base configuration $\mathcal{S}$ with $\mathcal{S} \subset \mathcal{T}$, the* sharpness of $\mathcal{S}$ *is defined as*

$$\mathbb{S}_{\mathcal{T}}(\mathcal{S}) = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{V \in \mathcal{V}_{\mathcal{T}}(\mathcal{S})} \left(\mathbb{E}^{loo}_{\mathcal{T}}(V) - \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S})\right)^2}$$

So, essentially the sharpness of some case base configuration mirrors how much isolated modifications to the case base (by including/removing a single case) influence the leave-one-out problem solving capabilities of $\mathcal{S}$.

4.2 Discussion of the Sharpness Measure

Using a root mean square definition instead of, for example, a sum over differences in Definition 5 serves two purposes. On the one hand, it allows for adequately assessing the level of sharpness of case base configurations $\mathcal{S}$ that are not local minima, i.e. where the inner difference is not guaranteed to be positive. On the other hand, it puts an emphasis on those case base configurations within the vicinity set $\mathcal{V}_{\mathcal{T}}(\mathcal{S})$ where the addition/removal of a single case has an above-average impact on the change in the error. It is also worth mentioning that calculating $\mathbb{S}_{\mathcal{T}}(\mathcal{S})$ is computationally costly with an effort of $O(|\mathcal{T}|^3)$ given that a simple linear retrieval is used in Equation 1.

While we have focused on a set of training cases $\mathcal{T}$ so far, we shall now put our attention more to an independent, held-out set $\mathcal{U}$ of test cases, i.e. $\mathcal{U} \subsetneq \mathcal{M}$ and $\mathcal{T} \cap \mathcal{U} = \emptyset$. Accordingly, our interest will be on the test error that some case base configuration $\mathcal{S}$ yields when its problem-solving capabilities are tested on $\mathcal{U}$. Hence, in analogy to Definition 3 we define:

Definition 6 (Test Error of a Case Base Configuration). *For a test set of cases $\mathcal{U} = \{c_1, \ldots, c_{|\mathcal{U}|}\}$ with each case $c_i = (p_i, s_i)$ consisting of a problem and*

solution part and for a given case base configuration $\mathcal{S}$, the test error *is defined as*

$$\mathbb{E}_{\mathcal{U}}(\mathcal{S}) = \frac{1}{|\mathcal{U}|} \cdot \sum_{i=1}^{|\mathcal{U}|} 1 - Correct(Adapt(Retrieve(\mathcal{S}, p_i), p_i), s_i)$$

Our conjecture is that the sharpness of a case base configuration is related to the testing performance of this case base. Thus, besides the actual value of the train error, the sharpness might help us in assessing the generalization capabilities of a case base configuration.

5 Empirical Evaluation

The first goal of our experimental evaluation is to empirically investigate the correlation between sharpness and test error, i.e. answering the question to what extent sharpness is suitable as a predictor of the generalization capability. In the second part of the evaluation, we aim at an empirical analysis of the four hill-climbing CBM variants proposed in Section 3.

We selected 21 classification domains from the UCI Machine Learning Repository [13] with varying amounts of case data, classes, and numbers and types of features. In all experiments, we split the available data set into two disjoint sets $\mathcal{T}$ and $\mathcal{U}$ where for the number of cases in the training set we focused on three settings ($|\mathcal{T}|$ being 50, 75, and 100, respectively). For k, as the number of nearest neighbors to be considered during retrieval we focused on $k = 1$ and $k = 3$.

5.1 Correlation Between Sharpness and Generalization

In machine learning, the training error is usually assumed to be strongly correlated to the error on an independent test set, except if overfitting has occurred. This general observation is also true for instance-based learning systems, expressing itself in a positive sample Pearson correlation coefficient $r_{x,y}$, where here x stands for the train error $\mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S})$ of a specific case base configuration and y for the test error $\mathbb{E}_{\mathcal{U}}(\mathcal{S})$ that $\mathcal{S}$ yields on an independent test set.

To this end, the interesting question is whether the sharpness $\mathbb{S}_{\mathcal{T}}(\mathcal{S})$ of a case base configuration $\mathcal{S}$ (cf. Definition 5) is also correlated with $\mathbb{E}_{\mathcal{U}}(\mathcal{S})$. In order to answer this question, we generated a large number of random case base configurations with random sizes by randomly adding any $c \in \mathcal{T}$ to $\mathcal{S}$ or not. For each domain, we processed 1000 such random case bases and determined $x = \mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S})$, $y = \mathbb{E}_{\mathcal{U}}(\mathcal{S})$, as well as sharpness values $z = \mathbb{S}_{\mathcal{T}}(\mathcal{S})$ (for brevity, we use x, y, and z as shorthand notation, subsequently). In so doing, we found that case base configurations with high sharpness tend to yield a higher test error, and vice versa. More specifically, the Pearson correlation $r_{z,y}$ is nearly identical to $r_{x,y}$ (see Table 1). In other words, the measure of sharpness introduced above is approximately as meaningful in assessing the generalization capability of a case base configuration as its leave-one-out train error.

	$\lvert\mathcal{T}\rvert = 50$			$\lvert\mathcal{T}\rvert = 75$			$\lvert\mathcal{T}\rvert = 100$		
	$r_{x,y}$	$r_{z,y}$	$r_{x+z,y}$	$r_{x,y}$	$r_{z,y}$	$r_{x+z,y}$	$r_{x,y}$	$r_{z,y}$	$r_{x+z,y}$
$k = 1$	0.608	0.624	0.675 (+0.066)	0.628	0.644	0.701 (+0.073)	0.647	0.646	0.697 (+0.050)
$k = 3$	0.652	0.617	0.687 (+0.034)	0.688	0.646	0.697 (+0.009)	0.706	0.685	0.728 (+0.021)

Table 1. Average Pearson correlations over all classification domains. For all settings examined, the correlation between train and test error can be improved (gain in brackets), when adding sharpness information to the train error.

Most interestingly, if we additively combine the train error $\mathbb{E}^{loo}_{\mathcal{T}}(\mathcal{S})$ and the sharpness $\mathbb{S}_{\mathcal{T}}(\mathcal{S})$ and determine the correlation of $\mathbb{E}_{\mathcal{T}} + \mathbb{S}_{\mathcal{T}}$ with $\mathbb{E}_{\mathcal{U}}$, i.e. $r_{x+z,y}$, we find that this is even higher than the correlation of both components alone (cf. Table 1). This is a strong indication that the sharpness can be helpful in estimating the generalization capabilities of the system. Figure 2 visualizes the gain in correlation for all the domains and all variations of k and $\lvert\mathcal{T}\rvert$ considered.

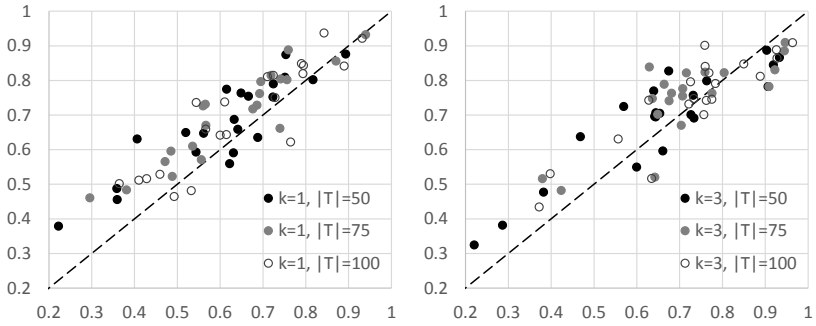

Fig. 2. Each data point represents the average over 1000 random case base configurations for one domain. The value on the abscissa refers to the average train-test correlation $r_{x,y}$, the value on the ordinate to the average sharpness-enhanced correlation $r_{x+z,y}$. Points above the identity function thus refer to runs where the incorporation of sharpness information brought about a better estimate of the generalization capabilities.

5.2 Hill-Climber Variants and Their Optima

Different optimization algorithms yield different minima. Depending on their nature they may tend to end up in rather flat or sharp local minimum of the error landscape. Given our observations reported in the preceding section, we conjecture that the sharpness of attained minima may help in evaluating the generalization capabilities of different algorithms. Besides this, we also want to empirically compare the performance of our proposed FHC and MHC variants with established CBM algorithms.

For this series of experiments, we applied all of the algorithms outlined above for each of the 21 classification domains, repeating this entire procedure 500 times for randomly re-initialized sets $\mathcal{T}$ of training cases (for both, $|\mathcal{T}| = 50$ as well as $|\mathcal{T}| = 100$, keeping $k = 1$ throughout).

First, we report the performance of the greedy case base editing schemes FHC and MHC in order to convey a feeling of their performance. As can be seen from Table 2, all four variants perform well, specifically $MHC_{\leq}$ features the best average test error while at the same time utilizing only less than one third of the cases given as input in $\mathcal{T}$. Domain identifiers are specified in a footnote[3]. We emphasize, however, that this comparison is of course not comprehensive; implementing other powerful case base editing algorithms from the literature (cf. Section 2) and matching our hill climbers' performance with those is an open topic for future work. Nevertheless, the numbers reported allow for concluding that FHC and MHC might be qualified as usable algorithms for case base maintenance.

Dom.	CNN		RNN		IB2		DROP1		DROP2		$FHC_{<}$		$FHC_{\leq}$		$MHC_{<}$		$MHC_{\leq}$	
A	.415	53.1	.422	49.6	.420	47.1	.430	**9.4**	.376	17.3	.355	22.2	**.350**	54.8	.360	**13.6**	**.340**	32.9
B	**.067**	22.0	.072	18.7	.090	19.4	.204	**13.2**	.078	21.5	.104	17.4	**.063**	55.0	.115	**11.3**	.085	26.1
C	.354	49.4	.361	43.9	.381	40.9	.355	**7.2**	.312	14.5	**.288**	14.7	.294	53.8	**.282**	**9.0**	.289	32.0
D	.344	47.6	.352	44.0	.356	43.3	.446	**9.5**	.346	18.2	.336	25.9	**.308**	67.0	.344	**14.5**	**.322**	42.7
E	.604	70.6	.606	64.1	.608	62.2	.625	**9.7**	.600	18.5	.597	19.8	**.592**	50.4	.598	**13.9**	**.592**	31.9
F	.258	41.5	.268	34.9	.278	36.1	.315	**15.6**	.238	**17.6**	.222	25.7	**.211**	68.1	.224	21.5	**.215**	37.9
G	.209	38.1	.222	30.7	.236	32.7	.250	**9.0**	.168	12.4	.164	19.1	.159	62.4	**.150**	**10.2**	**.156**	27.5
H	.380	53.3	.386	45.9	.398	41.1	.361	**8.2**	.323	13.7	**.305**	12.6	.328	53.5	**.298**	**6.5**	.319	28.3
I	**.365**	55.3	**.363**	41.4	.368	49.8	.538	17.5	.434	39.1	.466	23.8	.420	54.4	.487	**16.9**	.447	36.6
J	.278	43.6	.289	36.8	.305	33.2	.321	**7.7**	.229	13.4	**.208**	14.1	.209	46.2	.209	**10.2**	**.204**	23.4
K	.090	17.9	.094	14.6	.101	15.1	.147	**11.6**	.073	**12.7**	.073	15.3	**.067**	46.6	.076	13.0	**.068**	23.4
L	.298	41.3	.281	21.7	.359	36.8	.318	**9.5**	.294	33.1	**.236**	13.1	.243	50.4	**.236**	**9.7**	.240	27.4
M	**.321**	46.7	**.322**	43.1	.331	39.7	.441	**11.1**	.370	21.6	.382	19.1	.351	51.4	.381	**12.5**	.355	33.1
N	.356	51.6	.363	44.0	.373	39.3	.374	**7.7**	.333	14.3	.314	14.4	.317	50.5	**.312**	**10.3**	**.311**	26.8
O	.035	11.7	.036	10.1	.040	10.7	.114	**8.1**	**.035**	12.9	.038	13.4	**.026**	35.4	.043	**9.3**	.039	12.2
P	**.543**	73.2	**.542**	66.2	.563	62.9	.644	**18.2**	.606	27.4	.600	20.6	.583	56.4	.604	**16.3**	.589	36.9
Q	**.317**	48.4	**.323**	44.9	.335	41.5	.441	**12.0**	.372	22.7	.366	18.0	.325	59.1	.366	**9.5**	.344	40.1
R	.300	50.0	.306	43.0	.316	42.7	.465	**15.8**	.332	24.4	.351	27.3	**.309**	66.3	.355	**20.3**	**.327**	44.3
S	.324	49.2	.332	41.6	.345	40.2	.429	**11.2**	.316	16.6	.308	19.2	**.298**	59.4	.311	**14.0**	**.298**	36.4
T	.088	18.4	.094	14.9	.096	16.1	.137	**11.8**	.094	14.9	.074	16.6	**.068**	42.2	.078	**13.4**	**.071**	18.2
U	.584	72.0	.590	65.2	.592	64.5	.658	**12.2**	.580	**18.2**	.560	30.4	**.553**	62.3	.566	25.6	**.553**	42.3
μ_{50}	.311	45.5	.315	39.0	.328	38.8	.382	**11.3**	.310	19.3	.302	19.2	**.289**	54.5	.305	**13.4**	**.294**	31.5
μ_{100}	.305	43.6	.309	36.1	.326	36.5	.380	**7.9**	.290	16.4	.275	14.5	**.267**	56.7	.276	**8.5**	**.267**	31.0

Table 2. For each of the case base editing algorithms and for each of the considered domains two performance measures are given (averaged over 500 experiment repetitions, i.e. as many random sets $\mathcal{T}$ of train cases): the test error $\mathbb{E}_{\mathcal{U}}(\mathcal{S})$ of the learned case base configuration $\mathcal{S}$ as well as the achieved case base compactification in percent, i.e. $100\% \cdot \frac{|\mathcal{S}|}{|\mathcal{T}|}$. The table reports results for $|\mathcal{T}| = 50$ and $k = 1$, plus an average μ_{50} over all domains. Additionally, in the last line the average μ_{100} for $|\mathcal{T}| = 100$ is reported. For each domain, the two top-performing algorithms are highlighted. The first column contains domain identifiers (see footnote for plain text names).

[3] A-Balance, B-BanknoteAuth, C-Cancer, D-Car, E-Contraceptive, F-Ecoli, G-Glass, H-Haberman, I-Hayes, J-Heart, K-Iris, L-MammogrMass, M-Monks, N-Pima, O-QualBankruptcy, P-TeachAssistEval, Q-TicTacToe, R-UserKnowledge, S-VertebralCol, T-Wine, U-Yeast

Second, the hill climbing case base optimizers are designed to attain a local optimum of $\mathbb{E}^{loo}_{\mathcal{T}}$. Incidentally, some of our established case base editing algorithms do so, too, at least in some cases. Specifically, IB2 and CNN never converge to a local optimum of $\mathbb{E}^{loo}_{\mathcal{T}}$, but RNN (0.01%), DROP1 (4.3%), and DROP2 (24.7%) do so occasionally. The numbers in brackets refer to the share of the 21.000 experimental runs during which the respective algorithm attained an optimum. To this end, Figure 3 indicates that algorithms with a small sum of the train error and the sharpness (i.e. flat minima) correspond to smaller test error and, hence, better generalization than those that yield sharper minima. In particular, the train error alone is only of little use in predicting the test error (gray data points).

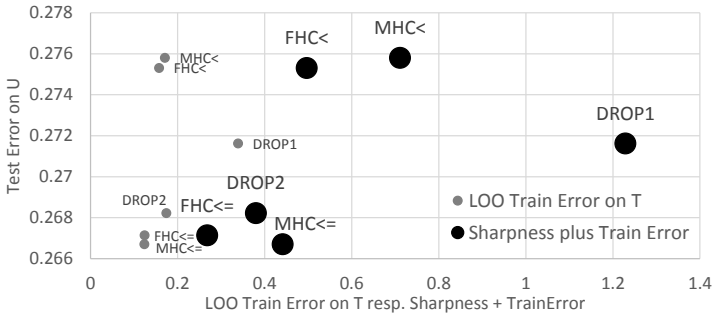

Fig. 3. Each data point stands for an average over the set of domains we considered and visualizes the correlation between the leave-one-out train error and the sharpness-enriched train error information with the generalization error measured on an independent test set $\mathcal{U}$. The plot refers to the setting of $k = 1$ with $|\mathcal{T}| = 100$, where for DROP1/2 only those runs were considered that yielded a case base configuration $\mathcal{S}$ that is a local optimum of $\mathbb{E}^{loo}_{\mathcal{T}}$.

6 Conclusion

In this paper, we have made a first attempt to investigate the presence and properties of sharp/flat minima in an error landscape of a case base maintenance scenario. We observed that sharp case base configurations feature poorer generalization properties than those that correspond to flat regions in the domain of the error function. Our analyses came along with two new case base maintenance procedures which, being hill-climbing optimizers, are by design attracted by local optima of the error function. We empirically found that their general performance is competitive in terms of case base competence and compactification and that the sharpness of local minima can be used to better predict the generalization error. An interesting avenue for future work is to design case base editing methods that incorporate sharpness information instantaneously, when adding or removing cases, and hence can be guided to attain flatter minima.

References

1. Aha, D., Kibler, D., Albert, M.: Instance-Based Learning Algorithms. Machine Learning 6, 37–66 (1991)
2. Brighton, H., Mellish, C.: On the Consistency of Information Filters for Lazy Learning Algorithms. In: Proc. of the Third European Conference on Principles of Data Mining and Knowledge Discovery. pp. 283–288. Prague, Czech Republic (1999)
3. Cover, T., Hart, P.: Nearest Neighbor Pattern Classification. IEEE Trans. Information Theory 13, 21–27 (1967)
4. Dinh, L., Pascanu, R., Bengio, S., Bengio, Y.: Sharp Minima Can Generalize for Deep Nets. In: Proceedings of the 34th International Conference on Machine Learning. pp. 1019–1028. JMLR.org, Sydney, Australia (2017)
5. Gates, G.: The Reduced Nearest Neighbor Rule. IEEE Trans. Information Theory 18(3), 431–433 (1972)
6. Hart, P.: The Condensed Nearest Neighbor Rule. IEEE Trans. Information Theory 14(3), 515–516 (1968)
7. Hochreiter, S., Schmidhuber, J.: Flat Minima. Neural Comp. 9(1), 1–42 (1997)
8. Keskar, N., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In: Proc of the 5th International Conference on Learning Representations. Toulon, France (2017)
9. Kibler, D., Aha, D.: Learning Representative Exemplars of Concepts: An Initial Case Study. In: Proceedings of the Fourth International Workshop on Machine Learning, pp. 24–30. Morgan Kaufmann (1987)
10. Leake, D., Wilson, D.: Categorizing Case-Base Maintenance: Dimensions and Directions. In: Proceedings of the 4th European Workshop on Case-Based Reasoning (EWCBR). pp. 196–207. Dublin, Ireland (1998)
11. Leake, D., Wilson, M.: How Many Cases Do You Need? Assessing and Predicting Case-Base Coverage. In: Proceedings of the 19th International Conference on Case-Based Reasoning (ICCBR 2011). pp. 92–106. Springer, London, UK (2011)
12. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the Loss Landscape of Neural Nets. In: Annual Conference on Neural Information Processing Systems 2018 (NeurIPS 2018). pp. 6391–6401. Montréal, Canada (2018)
13. Lichman, M.: UCI Machine Learning Repository (2013), archive.ics.uci.edu/ml
14. Lupiani, E., Craw, S., Massie, S., Juarez, J., Palma, J.: A Multi-Objective Evolutionary Algorithm Fitness Function for Case-Base Maintenance. In: Proceedings of the 21st International Conference on CBR. pp. 218–232. Springer, USA (2013)
15. Skalak, D.: Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. In: Proceedings of the 11th International Conference on Machine Learning. pp. 293–301. New Brunswick, NJ, USA (1994)
16. Smyth, B., McKenna, E.: Building Compact Competent Case-Bases. In: Proceedings of the Third International Conference on Case-Based Reasoning (ICCBR-99). pp. 329–342. Springer, Seeon Monastery, Germany (1999)
17. Smyth, B., McKenna, E.: Competence Guided Incremental Footprint-Based Retrieval. Knowledge-Based Systems 14(3-4), 155–161 (2001)
18. Wilson, D.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Trans. Systems, Man, and Cybernetics 2(3), 408–421 (1972)
19. Wilson, D., Martinez, T.: Reduction Techniques for Instance-Based Learning Algorithms. Machine Learning 38(3), 257–286 (2000)
20. Zhang, J.: Selecting Typical Instances in Instance-Based Learning. In: Proceedings of the 9th Intl. Workshop on Machine Learning. pp. 470–479. Aberdeen, UK (1992)